# gulp API以及常用插件

一、重要API

1.gulp.src(path, {options})

1.1 src方法是指定需要处理的源文件的路径，gulp.src返回当前文件流至可用插件

1.2 path：需要处理的源文件路径，必填

通配符路径匹配：

"src/main.js"：指定具体文件

"\*"：匹配所有文件夹（src/\*.js src下的所有js文件）

"\*\*"：匹配0个或者多个子文件夹 （src/\*\*/\*.js 包含src的0个或多个子文件夹下的js文件）

"{}"：匹配多个属性（src/{x,y}\*.js 包含s.js和y.js；src/\*.{jpg,png,gif} src下的所有jpg/png/gif文件）

"!"：排除文件（！src/main.js 不包括src下的a.js文件）

1.3 options：类型（可选），有3个属性buffer、read、base

buffer：类型Boolean，默认true，设置为false将返回file.content的流并且不缓冲文件，处理大文件时非常重要

read：类型Boolean，默认true，设置为false将不执行读取文件操作，返回null

base：类型String，设置输出路径以某个路径的某个组成部分为基础向后拼接

2.gulp.dest(path,[options])

2.1 指定处理完后文件输出的路径

2.2 path：指定文件输出路径或者定义函数返回文件输出路径亦可，必填

2.3 options：；类型（可选），有2个属性cwd、mode

cwd：类型String，默认process.cwd()前脚本的工作目录的路径 （当文件输出路径为相对路径将会用到）

mode：类型String，默认0777 指定被创建文件夹的权限

3.gulp.task(name,[deps],fn)

3.1 task定义一个gulp任务

3.2 name 指定任务的名称 必填

3.2 deps 该任务依赖的任务（被依赖的任务需要返回当前任务的事件流） 可选

3.3 fn 该任务调用的插件操作 必填

4.gulp.watch(path,opts,tasks/fn)

4.1 watch方法是用于监听文件变化，文件一修改就会执行指定的任务

4.2 path 需要处理的源文件匹配符路径 必填

4.3 opts 为一个可选的配置对象 通常不需要用到

4.4 tasks 需要执行的任务的名称数组 必填

4.5 fn 每个文件变化执行的回调函数 可选

二、常用插件

1.gulp-uglify（JS压缩）

1.1 安装 npm install gulp-uglify --save-dev

1.2 配置gulpfile.js

var gulp = require('gulp'),

uglify = require('gulp-uglify');

gulp.task('pressJS', function () {

gulp.src('src/js/test.js')

gulp.src(['src/js/test.js','src/js/index.js']) //多个文件

gulp.src(['src/js/\*.js','！src/lib/index.js']) //通配符匹配文件

.pipe(uglify())

.pipe(uglify({

mangle:{

expect:['require','exports',;$]/排除混淆关键字

}

}))

.pipe(uglify({

mangle: true,//类型：Boolean 默认：true 是否修改变量名

compress: true,//类型：Boolean 默认：true 是否完全压缩

preserveComments: 'all' //保留所有注释

}

}))

.pipe(gulp.dest('build/js'));

});

2.gulp-clean-css(CSS压缩) [gulp-minify-css]

2.1 安装 npm install gulp-clean-css --save-dev

2.2 配置gulpfile.js

var gulp = require('gulp'),

cleanCss = require('gulp-clean-css'),

rev = require('gulp-rev');

gulp.task('pressCSS', function () {

gulp.src('src/css/test.css')

gulp.src(['src/css/test.css','src/css/index.css']) //多个文件

gulp.src(['src/css/\*.css','！src/lib/index.css']) //通配符匹配文件

gulp.src(['src/css/\*.css','！src/lib/index.css']) //通配符匹配文件

.pipe(cssmin())

.pipe(cssmin({

advanced: false,//类型：Boolean 默认：true [是否开启高级优化（合并选择器等）]

compatibility: 'ie7',//保留ie7及以下兼容写法 类型：String 默认：''or'\*' [启用兼容模式； 'ie7'：IE7兼容模式，'ie8'：IE8兼容模式，'\*'：IE9+兼容模式]

keepBreaks: true,//类型：Boolean 默认：false [是否保留换行]

keepSpecialComments: '\*' //保留所有特殊前缀当你用autoprefixer生成的浏览器前缀，如果不加这个参数，有可能将会删除你的部分前缀

}))

.pipe(cssmin({

mangle: true,//类型：Boolean 默认：true 是否修改变量名

compress: true,//类型：Boolean 默认：true 是否完全压缩

preserveComments: 'all' //保留所有注释

}

}))

.pipe(rev()) //给css文件里引用文件加版本号（文件MD5）

.pipe(gulp.dest('build/css'));

});

3.gulp-htmlmin(压缩html文件) [gulp-minify-html]

3.1 安装 npm install gulp-htmlmin --save-dev

3.2 配置gulpfile.js

var gulp = require('gulp'),

htmlmin = require('gulp-htmlmin');

gulp.task('pressHtml', function () {

gulp.src('src/\*.html')

.pipe(htmlmin({

removeComments: true, //清除HTML注释

collapseWhitespace: true, //压缩HTML

collapseBooleanAttributes: true, //省略布尔属性的值 <input checked="true"/> ==> <input checked />

removeEmptyAttributes: true, //删除所有空格作属性值 <input id="" /> ==> <input />

removeScriptTypeAttributes: true, //删除<script>的type="text/javascript"

removeStyleLinkTypeAttributes: true, //删除<style>和<link>的type="text/css"

minifyJS: true, //压缩页面JS

minifyCSS: true //压缩页面CSS

}))

.pipe(gulp.dest('build/html'));

});

4.gulp-jshint(检查代码，打印报告信息)

4.1 安装 npm install gulp-jshint --save-dev

4.2 配置gulpfile.js

var gulp = require('gulp'),

jshint = require('gulp-jshint')

map = require('map-stream');//更详细的错误信息npm install map-stream --save-dev

var customerReporter = map(function(file,cb){

if (!file.jshint.success) {

//打印出错误信息

console.log('-----------------------------------------------');

file.jshint.results.forEach(function(err) {

if (err) {

console.log('在文件的第' + err.error.line + ' 行的第' + err.error.character + ' 列发生错误 ' + err.error.reason);

}

});

}

cb(null, file);

});

gulp.task('jshintJS', function () {

gulp.src('src/js/\*.js')

.pipe(jshint())

.pipe(jshint.reporter('YOUR\_REPORTER\_HERE'))

.pipe(customerReporter()) //输出更详细的错误信息

.pipe(rev()) //给css文件里引用文件加版本号（文件MD5）

.pipe(gulp.dest('build/css'));

});

5.gulp-concat(文件合并)

5.1 安装 npm install gulp-concat --save-dev

5.2 配置gulpfile.js

var gulp = require('gulp'),

concat = require('gulp-concat');

gulp.task('concat', function () {

gulp.src('src/js/\*.js')

.pipe(concat('app.debug.js')) //合并后的文件名

.pipe(gulp.dest('build/js'));

});

6.gulp-less(将less文件编译成css) sass和less用法相同

6.1 安装 npm install gulp-less --save-dev

6.2 配置gulpfile.js

var gulp = require('gulp'),

concat = require('gulp-less')

//确保本地已安装gulp-minify-css [cnpm install gulp-minify-css --save-dev]

cssmin = require('gulp-minify-css'),

//确保本地已安装gulp-sourcemaps [cnpm install gulp-sourcemaps --save-dev]

sourcemaps = require('gulp-sourcemaps'),

//当发生异常时提示错误 确保本地安装gulp-notify和gulp-plumber

notify = require('gulp-notify'),

plumber = require('gulp-plumber');

gulp.task('less', function () {

gulp.src('src/less/\*.less')

gulp.src(['src/less/test.less','src/less/index.less']) //多个文件

gulp.src(['src/less/\*.less','！src/lib/index.less']) //通配符匹配文件

.pipe(sourcemaps.init()) //当less有各种引入关系时，编译后不容易找到对应less文件，所以需要生成sourcemap文件，方便修改

.pipe(less())

.pipe(sourcemaps.write())

//.pipe(cssmin()) //压缩CSS 兼容IE7及以下需设置compatibility属性 .pipe(cssmin({compatibility: 'ie7'})) -->

.pipe(gulp.dest('build/css'));

});

gulp.task('testLess', function () {

gulp.src('src/less/\*.less')

.pipe(plumber({errorHandler: notify.onError('Error: <%= error.message %>')}))

.pipe(less())

.pipe(gulp.dest('src/css'));

});

gulp.task('testWatch', function () {

gulp.watch('src/\*\*/\*.less', ['testLess']);

});

7.gulp-imagemin(压缩图片)

7.1 安装 npm install gulp-imagemin --save-dev

7.2 配置gulpfile.js

var gulp = require('gulp'),

imagemin = require('gulp-imagemin'),

//确保本地已安装imagemin-pngquant [cnpm install imagemin-pngquant --save-dev]

pngquant = require('imagemin-pngquant');

gulp.task('imagemin', function () {

gulp.src('src/img/\*.{png,jpg,gif,ico}')

.pipe(imagemin({

optimizationLevel: 5, //类型：Number 默认：3 取值范围：0-7（优化等级）

progressive: true, //类型：Boolean 默认：false 无损压缩jpg图片

interlaced: true, //类型：Boolean 默认：false 隔行扫描gif进行渲染

multipass: true //类型：Boolean 默认：false 多次优化svg直到完全优化

//深度压缩图片

progressive: true,

svgoPlugins: [{removeViewBox: false}],//不要移除svg的viewbox属性

use: [pngquant()] //使用pngquant深度压缩png图片的imagemin插件

}))

.pipe(gulp.dest('build/img'));

});

8.gulp-livereload(当监听文件发生变化时，浏览器自动刷新页面)

8.1 安装 npm install gulp-livereload --save-dev

8.2 配置gulpfile.js

var gulp = require('gulp'),

concat = require('gulp-concat'),

livereload = require('gulp-livereload');

gulp.task('concat', function () {

gulp.src('src/js/\*.js')

.pipe(concat('app.debug.js')) //合并后的文件名

.pipe(gulp.dest('build/js'));

});

gulp.task('watch', function() {

livereload.listen(); //当concat发生变化的时候，自动更新页面

gulp.watch('src/js/\*\*/\*.js', ['concat']);

});

9.gulp-rev(根据静态资源内容，生成md5签名，打包出来的文件名会加上md5签名，同时生成一个json用来保存文件名路径对应关系)

9.1 安装 npm install gulp-rev --save-dev

9.2 配置gulpfile.js

var gulp = require('gulp'),

rev = require('gulp-rev'),//对文件名加MD5后缀

revCollector = require('gulp-rev-collector'); //路径替换

gulp.task('concat', function () {

return gulp.src('src/\*.css')

.pipe(rev())

.pipe(gulp.dest('dist'));

.pipe(rev.manifest()) // 生成一个rev-manifest.json

.pipe(gulp.dest('./rev')); // 将 rev-manifest.json 保存到 rev 目录内

});

gulp.task('rev', function() {

gulp.src(['./rev/\*.json', './application/\*\*/header.php']) //读取 rev-manifest.json 文件以及需要进行css名替换的文件

.pipe(revCollector()) // 执行文件内css名的替换

.pipe(gulp.dest('./application/')); //替换后的文件输出的目录

});

gulp.task('default', ['concat', 'rev']);

10.gulp-clean(删除文件)

10.1 安装 npm install gulp-clean --save-dev

10.2 配置gulpfile.js

var gulp = require('gulp'),

clean = require('gulp-clean');

gulp.task('clean', function() {

return gulp.src('build')

.pipe(clean()); //清除build文件

});